EXPERIMENT-10

Transactions & Concurrency Control

This document contains ready-to-run MySQL scripts demonstrating ACID transactions, atomic multi-row inserts, rollbacks on errors, and isolation simulation. Use it with Nimbus (https://bytexl.app/nimbus) or any MySQL client.

Setup — Create Table

```
DROP TABLE IF EXISTS FeePayments;

CREATE TABLE FeePayments (
  payment_id   INT        NOT NULL,
  student_name VARCHAR(100) NOT NULL,
  amount       DECIMAL(10,2) NOT NULL,
  payment_date DATE       NOT NULL,
  PRIMARY KEY (payment_id),
  CHECK (amount > 0)
) ENGINE=InnoDB;
```

Part A — Insert Multiple Fee Payments in a Transaction (Atomicity)

```
START TRANSACTION;

INSERT INTO FeePayments (payment_id, student_name, amount, payment_date)
VALUES
  (1, 'Ashish', 5000.00, '2024-06-01'),
  (2, 'Smaran', 4500.00, '2024-06-02'),
  (3, 'Vaibhav',5500.00, '2024-06-03');

COMMIT;

SELECT * FROM FeePayments ORDER BY payment_id;
```

This demonstrates Atomicity — all inserts succeed together as one unit.

Part B — Demonstrate ROLLBACK for Failed Payment Insertion

Option 1: Manual rollback after an error.

```
START TRANSACTION;

INSERT INTO FeePayments (payment_id, student_name, amount, payment_date)
VALUES (4, 'Kiran', 4000.00, '2024-06-04');

-- This insert fails (duplicate payment_id, negative amount)
INSERT INTO FeePayments (payment_id, student_name, amount, payment_date)
VALUES (1, 'Ashish', -100.00, '2024-06-05');

ROLLBACK;

SELECT * FROM FeePayments ORDER BY payment_id;
```

Option 2: Automatic rollback using a stored procedure.

```
DROP PROCEDURE IF EXISTS InsertMultiplePaymentsWithRollback;
DELIMITER //

CREATE PROCEDURE InsertMultiplePaymentsWithRollback()
BEGIN
  DECLARE EXIT HANDLER FOR SQLEXCEPTION
  BEGIN
    ROLLBACK;
  END;

  START TRANSACTION;

  INSERT INTO FeePayments (payment_id, student_name, amount, payment_date)
  VALUES (4, 'Kiran', 4000.00, '2024-06-04');

  INSERT INTO FeePayments (payment_id, student_name, amount, payment_date)
  VALUES (1, 'Ashish', -100.00, '2024-06-05');

  COMMIT;
END;
//
DELIMITER ;
```

```
CALL InsertMultiplePaymentsWithRollback();

SELECT * FROM FeePayments ORDER BY payment_id;
```

Part C — Simulate Partial Failure and Ensure Consistent State

```
DROP PROCEDURE IF EXISTS PartialFailureExample;
DELIMITER //

CREATE PROCEDURE PartialFailureExample()
BEGIN
  DECLARE EXIT HANDLER FOR SQLEXCEPTION
  BEGIN
    ROLLBACK;
  END;

  START TRANSACTION;

  INSERT INTO FeePayments (payment_id, student_name, amount, payment_date)
  VALUES (5, 'Nidhi', 3000.00, '2024-06-06');

  -- Invalid insert (NULL student_name)
  INSERT INTO FeePayments (payment_id, student_name, amount, payment_date)
  VALUES (6, NULL, 2500.00, '2024-06-07');

  COMMIT;
END;
//
DELIMITER ;

CALL PartialFailureExample();

SELECT * FROM FeePayments ORDER BY payment_id;
```

Even though the first insert is valid, the second fails — the entire transaction rolls back.

Part D — Verify ACID Compliance with Transaction Flow

1) Atomicity & Consistency: Shown in Parts A–C.

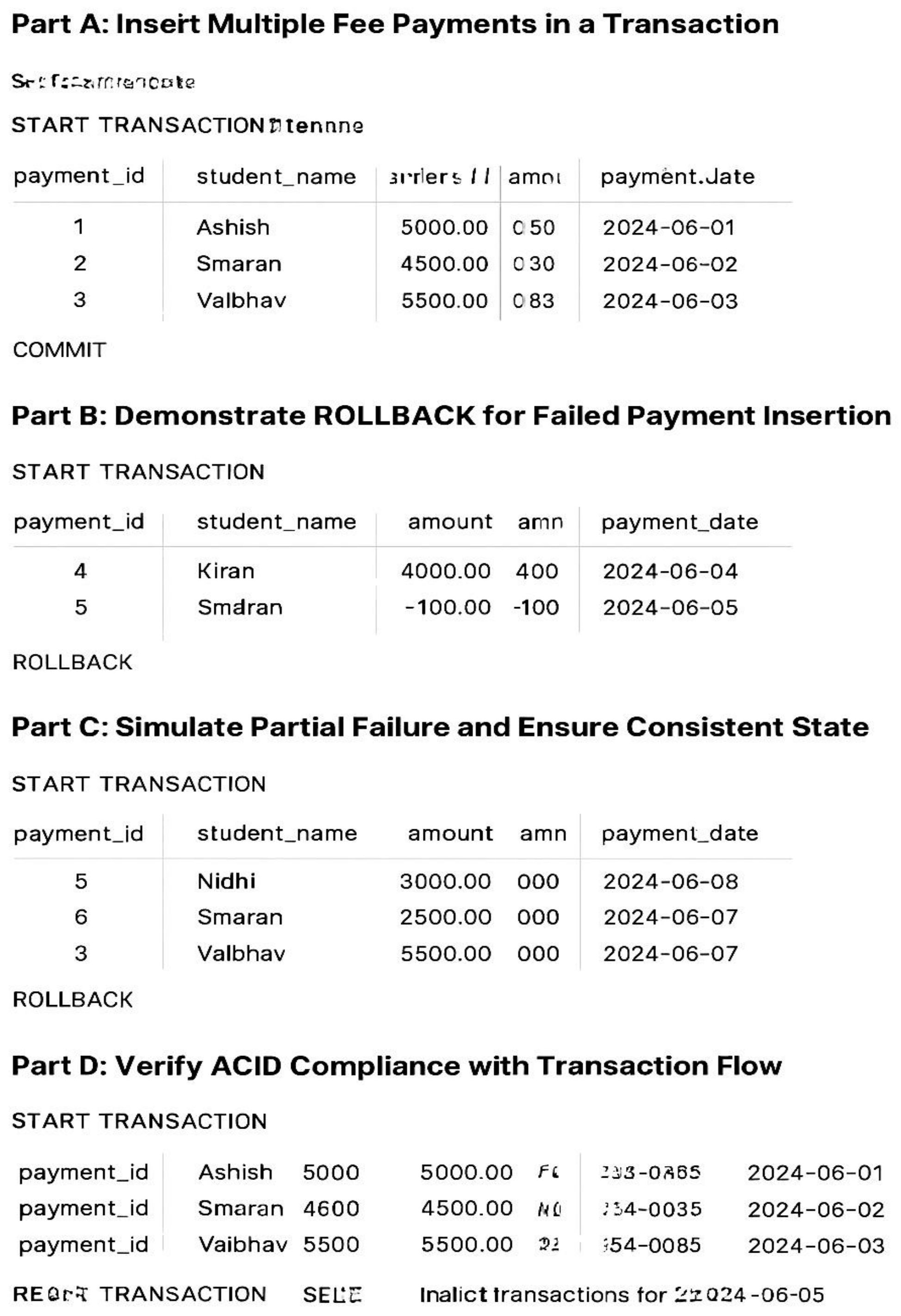
2) Isolation: Run in two sessions.

```sql
-- Session A
SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;
START TRANSACTION;
SELECT * FROM FeePayments WHERE payment_id = 2 FOR UPDATE;
UPDATE FeePayments SET amount = amount + 100 WHERE payment_id = 2;
-- COMMIT when done

-- Session B (run concurrently)
SET SESSION TRANSACTION ISOLATION LEVEL REPEATABLE READ;
START TRANSACTION;
UPDATE FeePayments SET amount = amount + 50 WHERE payment_id = 2;
COMMIT;
```

3) Durability: After COMMIT, data remains permanent even after restart.

OUTPUT :



Part A: Inseit Multiple Fee Payments in a Transaction

Sr: [illegible]

START TRANSACTION [illegible]tennne

payment_id	student_name	arrlers / /	amoı	payment.Jate
1	Ashish	5000.00	0 50	2024-06-01
2	Smaran	4500.00	0 30	2024-06-02
3	Valbhav	5500.00	0 83	2024-06-03

COMMIT

Part B: Demonstrate ROLLBACK for Failed Payment Insertion

START TRANSACTION

payment_id	student_name	amount	amn	payment_date
4	Kiran	4000.00	400	2024-06-04
5	Smdran	-100.00	-100	2024-06-05

ROLLBACK

Part C: Simulate Partial Failure and Ensure Consistent State

START TRANSACTION

payment_id	student_name	amount	amn	payment_date
5	Nidhi	3000.00	000	2024-06-08
6	Smaran	2500.00	000	2024-06-07
3	Valbhav	5500.00	000	2024-06-07

ROLLBACK

Part D: Verify ACID Compliance with Transaction Flow

START TRANSACTION

payment_id	Ashish	5000	5000.00	[illegible]	[illegible]3-0865	2024-06-01
payment_id	Smaran	4600	4500.00	[illegible]	[illegible]4-0035	2024-06-02
payment_id	Vaibhav	5500	5500.00	[illegible]	[illegible]54-0085	2024-06-03

RE[illegible] TRANSACTION SELE[illegible] Inalict transactions for [illegible]024-06-05